JAVA ASSIGNMENT

(Ques 1 - 5 theory)
(Ques 6 - 7 GitHub link)

Ques1 Identify and list down key differences among JDK, JRE and JVM.

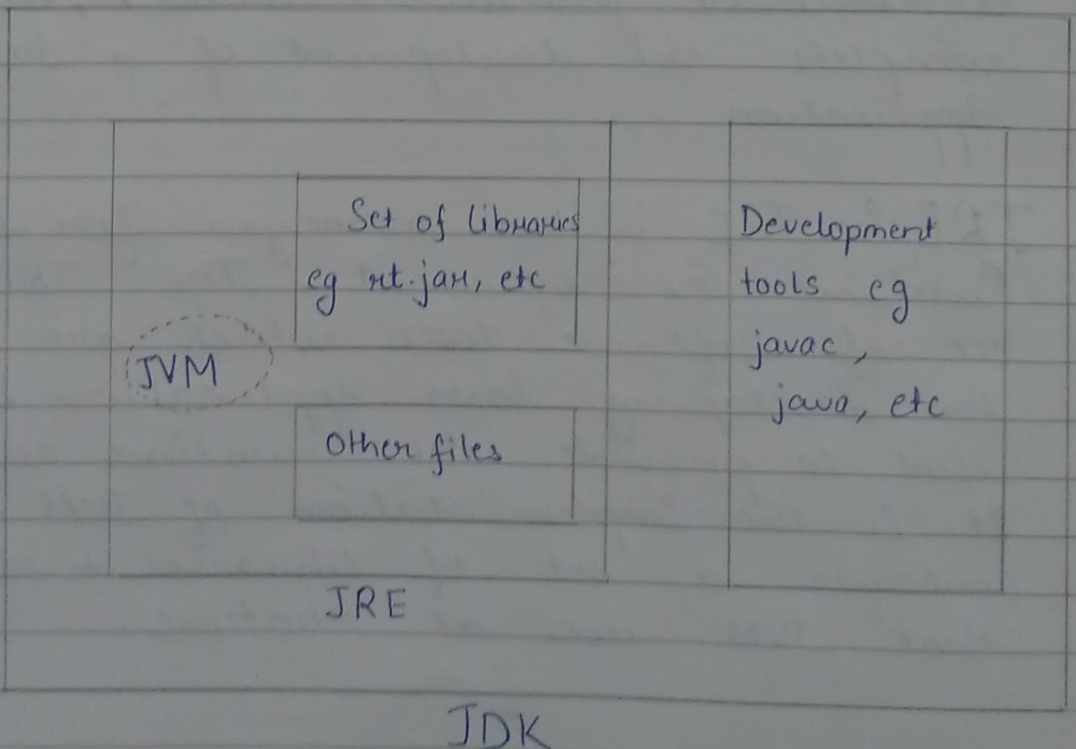
Ans1 JDK: JDK is an acronym for Java Development Kit. The JDK is a software development environment which is used to develop Java applications and applets.
It contains a private Java Virtual Machine (JVM) and a few other resources such as an interpreter / loader (java), a compiler (javac), an archiver (jar), a document generator (Javadoc), etc to complete the development of a Java Application.

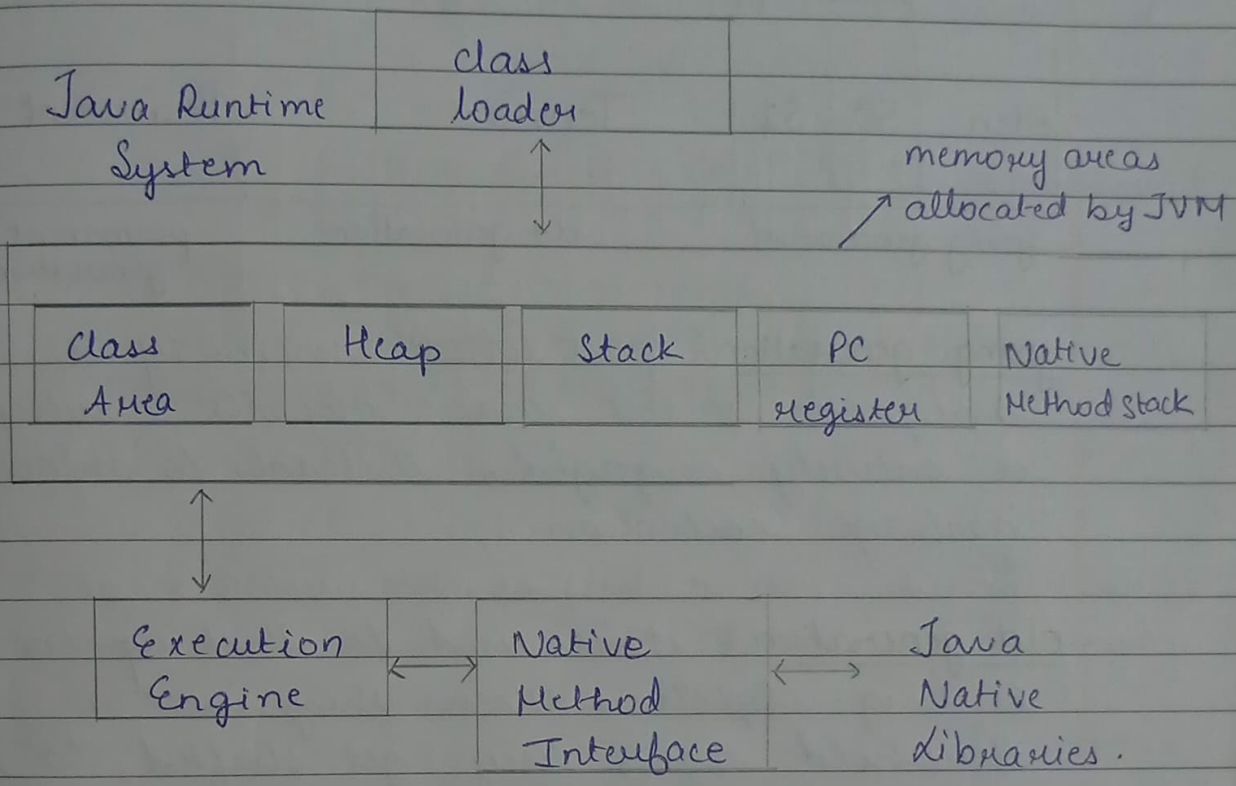
JRE: JRE is an acronym for Java Runtime Environment. The JRE is a set of software tools which are used for developing Java Application. It is used to provide the runtime environment. It is the implementation of JVM. It contains a set of libraries + other files that JVM uses at runtime.

JVM: JVM (Java Virtual Machine) is an abstract machine. It is called virtual machine because it doesn't physically exist. It is a specification that provides a runtime environment in which Java byte code can be executed. JVM, JRE and JDK are platform dependent because the configuration of each OS is different from each other.

The JVM performs the following main tasks -

→ Loads code
→ Verifies code
→ Executes code
→ Provides runtime environment

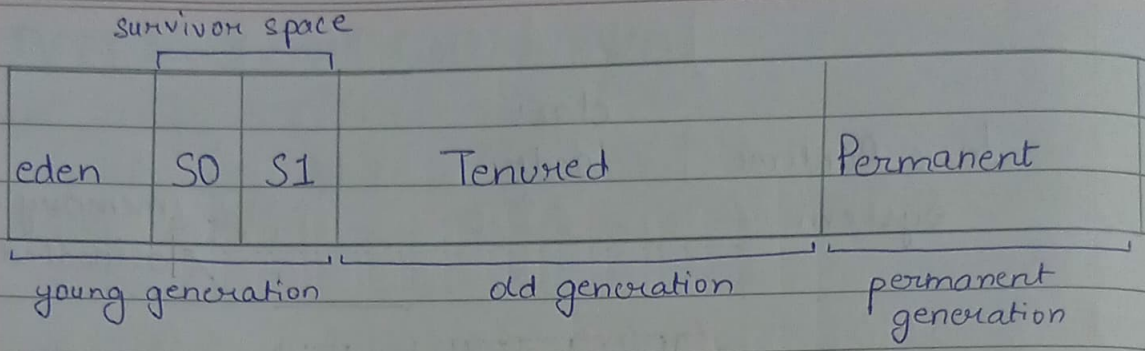




Ques2 Explain the garbage collection process in java with diagrams covering various generations where the memory is managed.

Ans2 Garbage is a heap of unreferenced objects which occupy memory space without any need. In Java, distruction of objects from memory is done automatically by JVM.

The stack of memory is broken down into smaller generations -



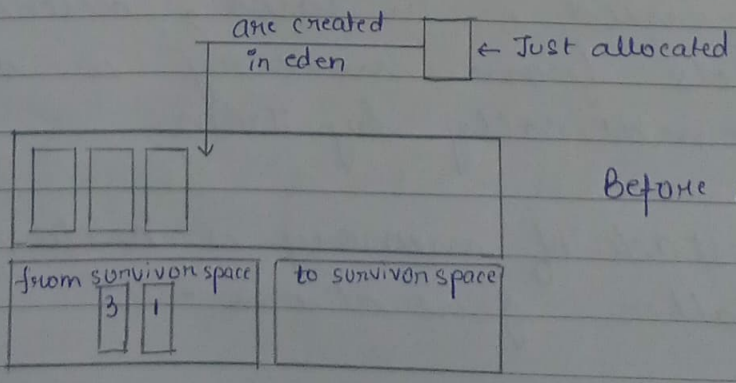
young generation: space where memory is assigned to all new objects, when it is entirely engaged, it leads to minor garbage collection.

old generation: it is used to stock up long existing objects. When they meet a threshold age, they get shifted to permanent generation.

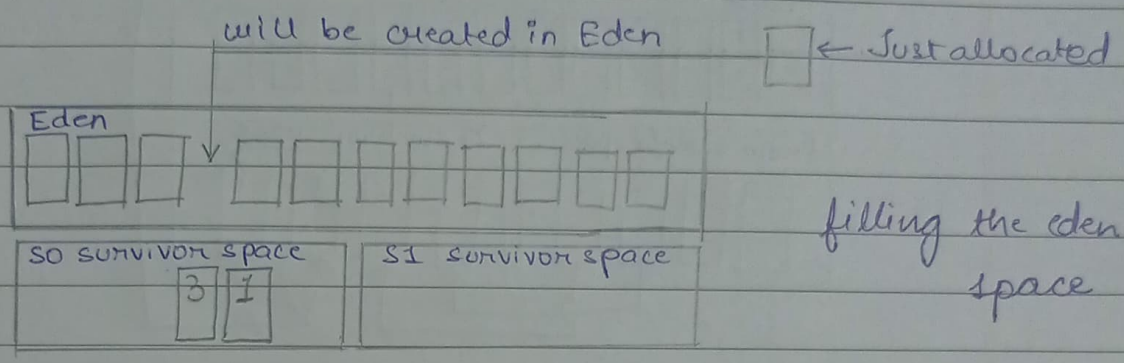
permanent generation: contains metadata required by the JVM to describe the class and method used.

GARBAGE COLLECTION PROCESS IN JAVA

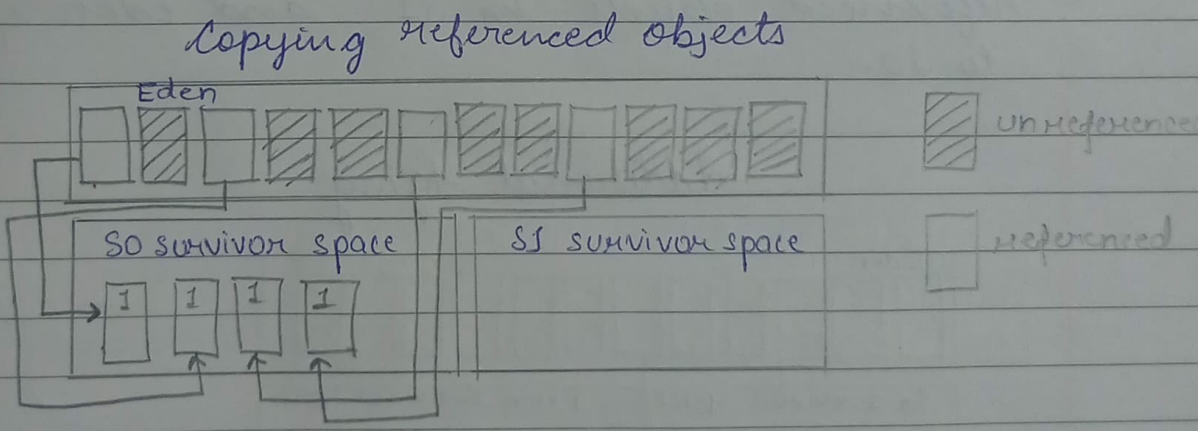
1. Whenever a new object is formed, it gets its memory assigned to eden space.



2. Upon filling of eden space, it is accompanied with a minor garbage collection.

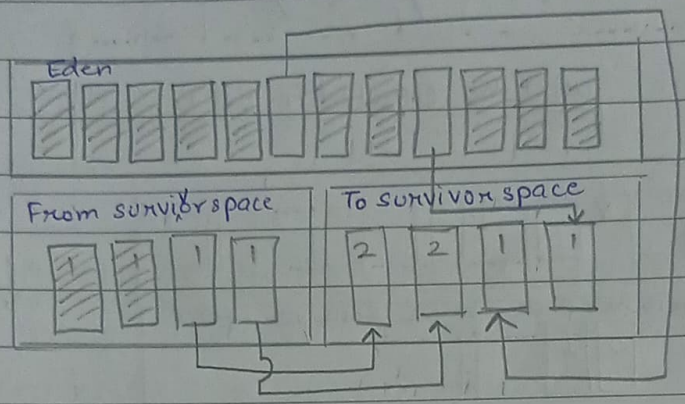


3. Reference objects are shifted to the first survivor space, the unreferenced objects are dumped and eden space is cleared.



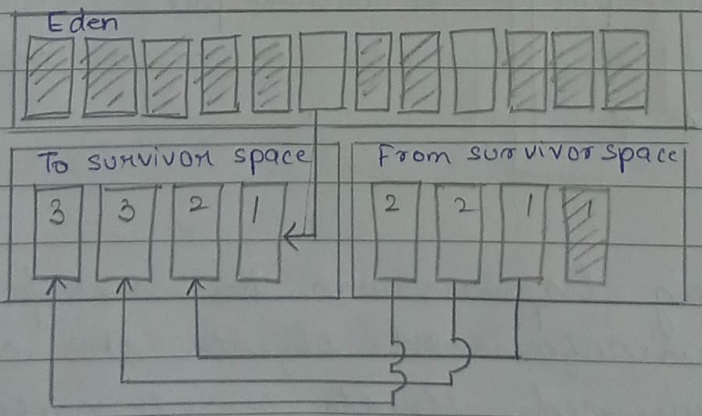
4. Same thing happens over the next period of time, unreferenced objects are cleared and referenced objects are transferred to survivor space (S1). In addition to that both eden space and survivor space are cleared after moving referenced objects of survivor space S0 to S1.

Object Aging



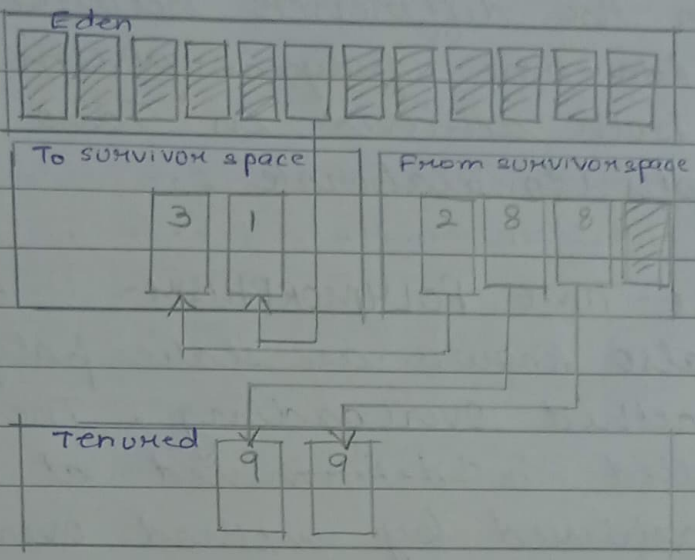
5. The survivor space ~~switches~~ switches at next level of minor GC. Eden spaces are cleared after the movement of referenced objects in S1 and eden space to S0.

Additional Aging

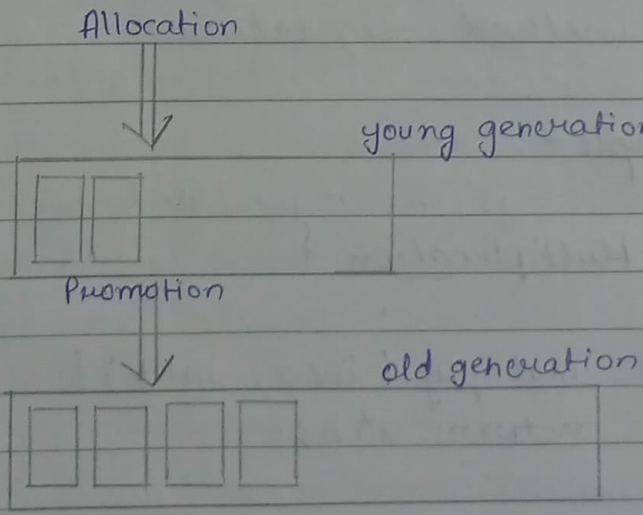


6. Now when objects arrive at a minimum threshold age, they are popped up to the old generation.

Promotion



7. With the continuation of minor garbage collection the objects are continously popped up to old generation.



8. Ultimately, a major garbage collection is performed, which cleans up and compacts that space of old generation.

Ques3 Explain the difference between compile and run-time polymorphism. How are both achieved in Java? Give examples to elaborate.

Ans3 COMPILE-TIME POLYMORPHISM-

It is also known as static polymorphism or method overloading. The method to be called is determined at compile-time. It is achieved by method overloading.

→ The method name is same but parameter's type, number or order is different.

→ Decision is made by the compiler, based on the method signature.

Example-

```
class Multiplication {

    int multiply (int a, int b) {
        return a*b;
    }

    double multiply (double a, double b) {
        return a*b;
    }

    int multiply (int a, int b, int c) {
        return a*b*c;
    }
}
```

```
public class Main {
    public static void main(String[] args){

        Multiplication m = new Multiplication();

        //calling multiply (int, int)
        System.out.println( m.multiply(1,2));

        //calling multiply (double, double)
        System.out.println( m.multiply(3.7, 5.8));

        //calling multiply ( int, int, int )
        System.out.println( m.multiply(2, 3, 5));
    }
}
```

RUNTIME POLYMORPHISM –

It is also known as dynamic polymorphism or method overriding. The method to be called is determined at runtime based on the actual object type.
It is achieved by method overriding.

→ A subclass provides a specific implementation for a method that is already defined in its superclass.

→ Decision is made by JVM during runtime based on the object's actual type (not the reference type)

Example -

```
class Vehicle {
    void start() {
        System.out.println("Vehicle is starting.");
    }
}

class Car extends Vehicle {
    @Override
    void start() {
        System.out.println("Car is starting.");
    }
}

class Bike extends Vehicle {
    @Override
    void start() {
        System.out.println("Bike is starting.");
    }
}

public class Main {
    public static void main(String[] args) {

        Vehicle vehicle;

        vehicle = new Car();
        vehicle.start(); // Car is starting.

        vehicle = new Bike();
        vehicle.start(); // Bike is starting.
    }
}
```

Ques4 What's an Exception?

a) Under what circumstances should a developer throw an exception?

b) Should a thrown exception always be caught?

c) What's the difference between checked and unchecked exceptions?

d) Write the code for defining a custom exception MyCustomException.

Ans4 @ In Java, runtime errors are thrown as exceptions.

An exception is an object that represents an error or a condition that prevents execution from proceeding normally.

If an exception is not handled, the program will terminate abnormally.

a) A developer should throw an exception under the following circumstances –

→ Invalid input : when the method receives invalid or unexpected input that it cannot process

→ Illegal state : when the application is in an invalid or unexpected state.

→ Resource Issues : when a required resource is unavailable or inaccessible

→ Program logic Errors : to indicate violations of program logic, such as invalid

method calls or conditions that should not occur

b) No, a thrown exception does not always need to be caught, but it must be handled either by catching or by propagating to the calling method.

c)

CHECKED EXCEPTIONS	UNCHECKED EXCEPTIONS
→ They are checked at compile time	These occur at runtime and are not checked at compile time
→ Must be either caught or declared using throws.	No need to catch or declare them.
→ Subclasses of Exception (except RuntimeException)	Subclasses of RuntimeException
→ Situations that the program can recover from.	Programming logic errors or unexpected issues
→ Example: IOException, SQLException, ClassNotFoundException	Example: NullPointerException, ArithematicException, ArrayIndexOutOfBoundException

d) public class MyCustomException extends Exception{

```
    public MyCustomException(){
        super("Exception");
    }
    public MyCustomException(String msg){
        super(msg);
    }
}
```

Ques5 How would you explain the difference between Concurrency and Parallelism?

Ans5 CONCURRENCY: means an application is handling multiple tasks at the same time. It helps reduce the response time of the system by using a single processor. Although it looks like ~~tasks~~ tasks are being processed in parallel, they actually take turns. The system does not fully complete one task before starting another.

Concurrency works by quickly switching between tasks on the CPU. This creates the impression of parallel work, even though it is not truly simultaneous.

PARALLELISM: is when tasks are divided into smaller parts and processed at the same time. It improves speed and performance by ~~tising~~ using multiple processors. This makes it possible to run tasks simultaneously.

Parallelism overlaps the processing of tasks across multiple CPUs or processors. Unlike concurrency, which overlaps input/output and CPU tasks to improve speed, ~~parallel~~ parallelism uses actual simultaneous processing to handle tasks faster.